SISTEMI OPERATIVI

ESERCIZIO N. 1 del 10 GENNAIO 2001

In un negozio sportivo lavorano **C commessi** e un **supervisore**. Il negozio è frequentato da **clienti normali** e **clienti che chiedono la sostituzione** di articoli acquistati precedentemente. Il negozio ha una capacità massima di **CAP** clienti. Ogni cliente normale è servito da un commesso, mentre per servire un cliente che chiede una sostituzione è necessaria la presenza di un commesso e del supervisore. I clienti normali (e solo quelli normali), terminato l'acquisto, vanno alla cassa a pagare, mettendosi in coda nel caso in cui sia già occupata.
Si implementi una soluzione usando il costrutto monitor per modellare il **negozio** e i processi per modellare il **clienti** (si considerino i commessi e il supervisore delle risorse) e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program shopping

const     C = ...; { numero di commessi }
const     CAP = ...; { capacità del negozio }
typetipo = (N, S);  { cliente Normale e per Sostituzione }

type cliente = process (t: tipo)
begin
      negozio.entra;
      <guarda cosa c'è di bello>
      negozio.chiedi_commesso(t);
      <acquista o cambia>
      negozio.vai_alla_cassa(t);
      <paga o passa>
      negozio.esci (t);
end

type negozio_sportivo = monitor

{ variabili del monitor }
var  commessi_occupati : integer;
      { numero di commessi occupati }
      posti_liberi : integer;
      { posti liberi nel negozio }
      super_occupato : boolean;
      { stato del supervisore }
      cassa_occupata : boolean;
      { stato della cassa }
      coda_fuori : condition;
      { coda su cui sospendere prima di entrare }
      coda_dentro : array[tipo] of condition;
      { clienti che aspettano un commesso }
      coda_cassa : condition;
      { clienti che aspettano di pagare }
```

```
procedure entry entra
begin
	if posti_liberi = 0 then
	{ controllo se c'è un posto libero }
		coda_fuori.wait;
	posti_liberi -- ; { sono entrato }
end

procedure entry chiedi_commesso (t : tipo)
begin
	if t = N then { cliente normale }
	begin
		if commessi_occupati = C then
		  coda_dentro[t].wait;
		commessi_occupati ++ ;
	end
	else { tipo = S }
	begin
		{ controllo anche il supervisore }
		if commessi_occupati = C or super_occupato then
		  coda_dentro[t].wait;
		commessi_occupati ++ ;
		super_occupato = true;
	end
end

procedure entry vai_alla_cassa(t: tipo)
begin
	commessi_occupati -- ;
	if t = S then
	  super_occupato := false;
	{ risveglio un cliente dando la priorità a quelli normali }
	if coda_dentro[N].queue then coda_dentro[N].signal;
	else
	  if coda_dentro[S].queue and not super_occupato then
		coda_dentro[S].signal;
```

```
      if t = N then
      begin
        if cassa_occupata then
        { controllo se la cassa è libera }
            coda_cassa.wait;
        cassa_occupata = true;
      end
end

procedure entry esci (t : tipo)
begin
      if t = N then
      begin
            cassa_occupata = false;
            if coda_cassa.queue then coda_cassa.signal;
      end

      posti_liberi ++ ;
      { risveglio un solo cliente }
      if coda_fuori.queue then
            coda_fuori.signal;
end

begin { inizializzazione delle variabili }
      commessi_occupati := 0;
      posti_liberi := CAP;
      super_occupato = false;
      cassa_occupata = false;
end

var  negozio: negozio_sportivo; { il nostro monitor }
      n1, n2, ... : cliente (N);
      s1, s2, ... : cliente (S);

begin end.
```